Rotor: Bringing .NET to FreeBSD

Stephen R. Walli
Program Manager, Rotor Project

Outline

- Project Goals
- The License
- The ECMA standards work
- Rotor architecture
- Platform adaptation layer
- Base Class Library Selection

What is Rotor?

- Rotor is a reference implementation of the ECMA CLI and C# specs, plus key SDK tools, delivered as source under a license with generous terms for non-commercial purposes.
- It will build and run on Windows and FreeBSD

Why Are We Building Rotor?

- Key reason: support of proposed ECMA standard
  - Proof point for alternate platforms
  - Approachable "guide" for teachers, experimenters, and other commercial implementers
- ECMA standard relates compact framework, .NET framework, and third-party implementations
- Why bootstrap alternate CLI implementations?
  - Establish new runtime platform around web services
  - Interop will be a key to customer satisfaction
  - "Interop" is not "cross platform" (WORA is non-goal)

Shared Source CLI License

- Part of Microsoft's shared source framework
  - Multiple licenses offer flexibility and choice
  - Maintains the intellectual property rights needed to support strong software ecosystem
  - Adopts beneficial aspects of open source models, especially in the areas of source transparency and community
- One representative license will be for the shared source CLI, C#, and ECMAScript implementations:
  - Neither published nor final, but similar to published WinCE license
  - Feedback welcome!
- Goals of license design
  - Long-term investment of intellectual property into intellectual commons
  - Very liberal rights w.r.t. non-commercial derivatives, to increase educational access to seed the future
  - "Taintless" viewing of source when used as an implementation guide

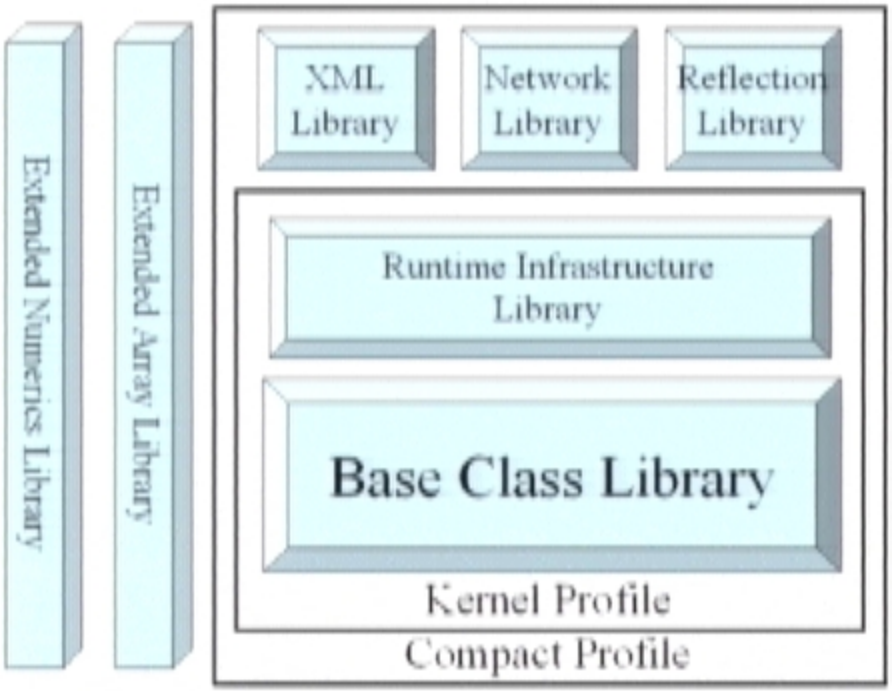
ECMA Submissions

- Partition I: Concepts and Architecture
  - Includes common type system
- Partition II: Metadata Definition and Semantics
  - Includes ilasm syntax, file format, PE and metadata
- Partition III: CIL Instruction Set
  - Includes instruction validation and verification rules
- Partition IV: Profiles and Libraries
  - Describes partitioning and common material
- Partition V: Annexes
  - Validation, verification, design guidelines, etc.
- C# Programming Language
  - Complete language specification, CLI is runtime for C#

Detail: Base Class Libraries

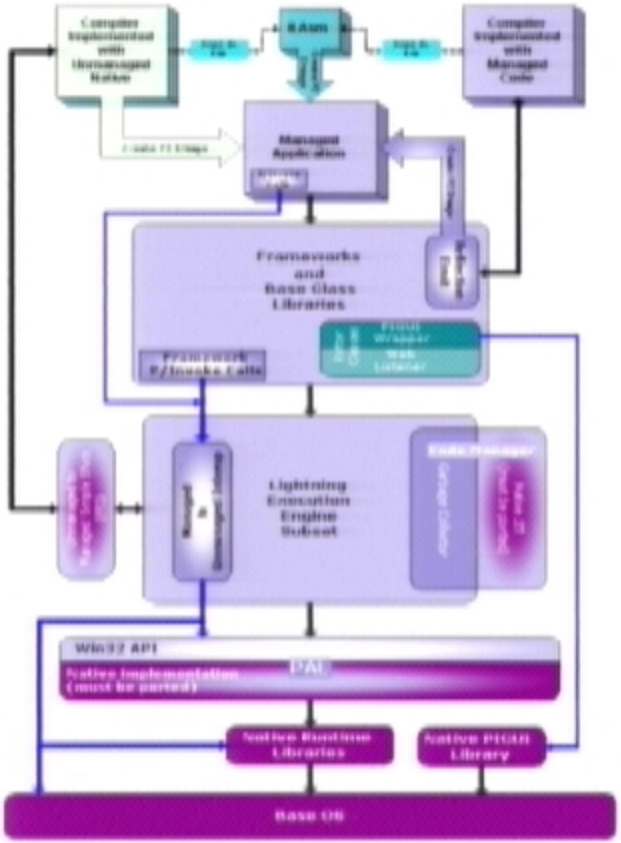
- Abstract OS Interface
  - Networking, Security, Standard I/O, Threading
- Common Programming Library
  - Common Data Types, Advanced Data Types, Collections, Extended Numerics, Regular Expressions, Serialization
- High-Level Programming
  - Asynchronous Programming, Globalization, Remoting, XML, Advanced XML
- EE Functionality
  - GC, Hosting, NonCLS, Reflection, Unmanaged
- Miscellaneous
  - Kernel, Basic Language Support, Development Time

Map of Libraries within Profiles



Interesting Rotor Components

- PAL
- Alternate JIT
- Simpler GC
- Hosting and calls into managed code
- Platform Independent GUI



PAL

- PAL implements platform support in a generic way
  - Boot loader
  - Threading support
  - Networking
  - Timers
  - File system
- PAL APIs preserve Win32 aspect but are chosen to facilitate porting to Unix platforms
  - Designed to minimize changes in DNoW code base
- PAL used by the
  - CLR
  - FX classes via P/Invoke
  - Unmanaged tools
- About 200 Win32 APIs will be implemented
  - Eliminating WinForms simplified it considerably

PIGUI Plan

- Many competing solutions within MS
  - Starlite, UV, EyeOpener, DUser
  - But none have critical mass at this point
  - None have any mindshare outside MS
- Many competing external PIGUI libraries
  - Major Linux battleground
    - GNOME on top of GTK+ vs KDE on top of Qt
    - Very controversial area
  - All external solutions have significant issues
    - wxWindows is C++ based
      - All C++ based libraries will be difficult to wrap
    - Qt has licensing issues on Windows
    - GTK+ has ugly programming model and no Win32 option
    - Tk is very retro look and feel and possibly slow
      - No combo box, no tree widget
- Current plan is to wrap Tk
  - Fits with ubiquity goal
  - Will be positioned as only one option/sample

Rules for Cutting

- Started with everything in DNoW and then cut
  - Didn't build up from ECMA standards
- Cut criteria:
  - Remove trade secret or other sensitive IP
    - In most cases there are alternate low-IP code bases
  - Remove Windows-specific functionality that can't feasibly be ported in v1
  - Remove features that are beyond the basics

Significant Cuts and Exceptions

- **COM interop cut**
  - COM not common on target platforms
  - Expensive cut since it broadly impacts source
- **Windows Forms cut**
  - Too dependent on Win32 to become basis of platform indep. GUI
- **ASP.NET cut**
  - But System.Web.Services* in Rotor
- **MS-specific areas of data classes cut**
  - SQL Server and OLEDB providers are cut
  - Other areas under investigation
- **VS.NET support cut**
  - No cross-dev scenario
    - Debug APIs in Rotor are subset so can't use VS debugger against Rotor
      - Use CorDBG instead
  - All *.Designer.* classes cut
  - VB and VC compiler will not be ported
  - But C# and JScript will be ported
- **Transparent loading and IJW support**
  - Rotor will not load IJW images
  - Users must use RunCom to execute managed apps

Availability

- Q1/Q2 2002 for beta release
- Q3/Q4 2002 for final release
- Community build exercise around the release

Additional Resources

- Project Web site:
  http://rotor
- ECMA specs:
  http://msdn.microsoft.com/net/ecma
- Whitepaper and FAQ:
  http://www.microsoft.com/net/whitepapers.asp
- Microsoft commercial C# and CLR beta:
  http://msdn.microsoft.com/downloads
- Shared source info:
  http://www.microsoft.com/sharedsource
  http://www.microsoft.com/windows/embedded/ce

Questions?

Loader.NET: Software On Demand

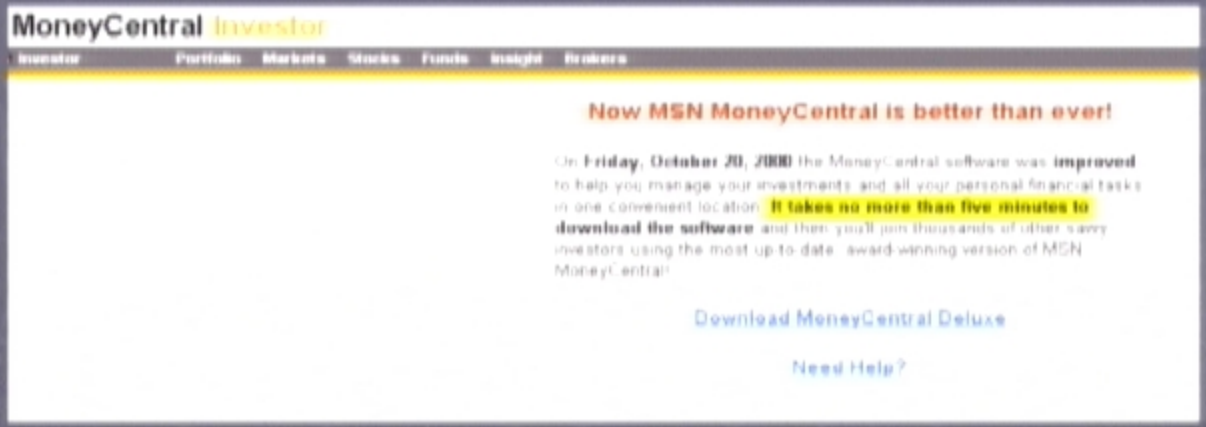
Hoi Vo
PPRC Perf Optimization
MS Research

Agenda

- Introduction
- Our current effort: Loader.NET
- Technical Challenges
- Case studies
- Where we go from here…

Introduction

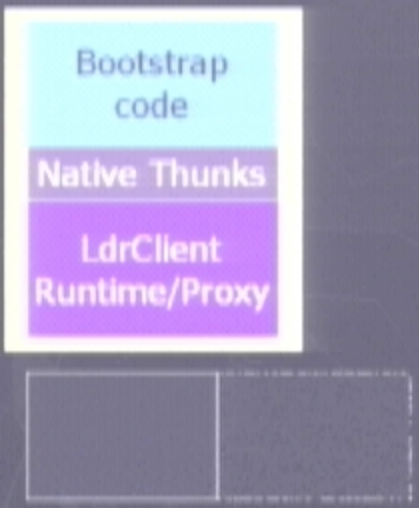
- MSN Money Central is a 1.5 MB online app
- 550KB compressed file to download



.NET challenges

- Memory hierarchy now includes
  - Network latencies
  - Downlink bandwidth
- No unified solution for software delivery
  - Subscription model does not handle legacy apps
  - No personalization mechanism defined
- Optimization needs profile data
  - Difficult to predict correct workload
  - No current infrastructure for optimization on actual live system

Loader.NET

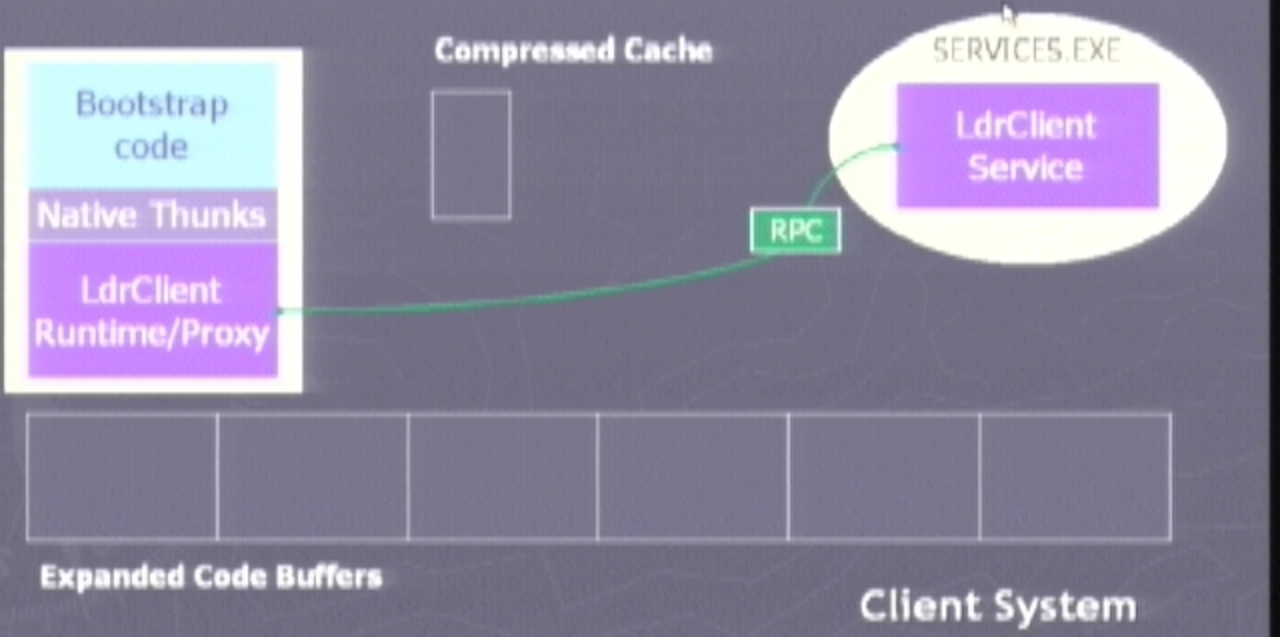


Client System

Loader.NET


Bootstrap code
Native Thunks
LdrClient Runtime/Proxy
Compressed Cache
SERVICES.EXE
LdrClient Service
Expanded Code Buffers
Client System

Loader.NET


Compressed Cache
SERVICES.EXE
LdrClient Service
RPC
Bootstrap code
Native Thunks
LdrClient Runtime/Proxy
Expanded Code Buffers
Client System

Technical Challenges

- Debugging
- Disconnection
- Security
- Scalability
- Data files / Resources
- Offline application mode
- No CodeView
- Debugging
- Debugging
- Debugging...

Technical Challenges

- Debugging
- Disconnection
- Security
- Scalability
- Data files / Resources
- Offline application mode
- No CodeView
- Debugging
- Debugging
- Debugging...

Sparse Binary Concept

Sparse Binary Concept

- Solves the debugging problem
  - Same layout as original binary
  - Can leverage from same PDB
  - Majority of code/data in sparse binary is removed
  - Filled mostly with 0 increases compression ratio

Sparse Binary Concept

- Solves the debugging problem
  - Same layout as original binary
  - Can leverage from same PDB
  - Majority of code/data in sparse binary is removed
  - Filled mostly with 0 increases compression ratio
- Downsides:
  - No dynamic optimization support
  - Stub PE still significant
  - Require larger patch for hotfix or upgrade

Sparse Binary Concept

- Solves the debugging problem
  - Same layout as original binary
  - Can leverage from same PDB
  - Majority of code/data in sparse binary is removed
  - Filled mostly with 0 increases compression ratio
- Downsides:
  - No dynamic optimization support
  - Stub PE still significant
  - Require larger patch for hotfix or upgrade

Sparse Binary Concept

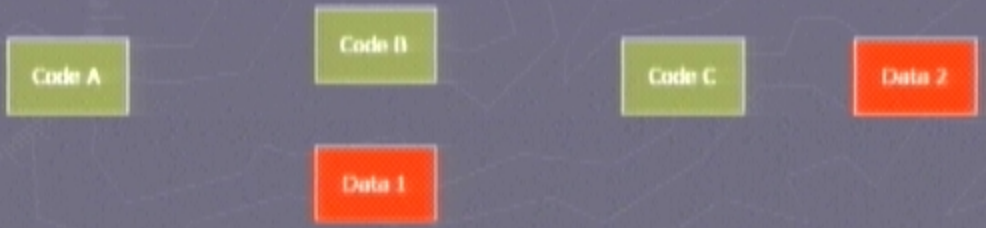
- Solves the debugging problem
  - Same layout as original binary
  - Can leverage from same PDB
  - Majority of code/data in sparse binary is removed
  - Filled mostly with 0 increases compression ratio
- Downsides:
  - No dynamic optimization support
  - Stub PE still significant
  - Require larger patch for hotfix or upgrade

Handling Data Blocks

- Issues:
  - Multiple references from different code paths
  - R/W data blocks require stricter sync during decomp
  - Too large data arrays require special handling
  - Special PE data blocks cannot be streamed
  - Singly referenced data blocks detection

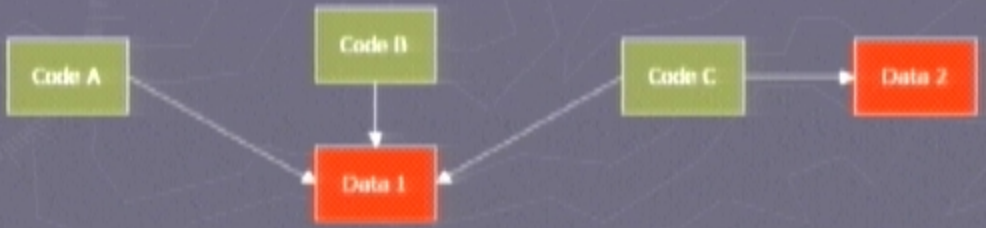
Handling Data Blocks

- Issues:
  - Multiple references from different code paths
  - R/W data blocks require stricter sync during decomp
  - Too large data arrays require special handling
  - Special PE data blocks cannot be streamed
  - Singly referenced data blocks detection



Handling Data Blocks

- Issues:
  - Multiple references from different code paths
  - R/W data blocks require stricter sync during decomp
  - Too large data arrays require special handling
  - Special PE data blocks cannot be streamed
  - Singly referenced data blocks detection



Handling Data Blocks

- Issues:
  - Multiple references from different code paths
  - R/W data blocks require stricter sync during decomp
  - Too large data arrays require special handling
  - Special PE data blocks cannot be streamed
  - Singly referenced data blocks detection

Download Cache Optimization

- Use BBT profile data for static layout
- Cached funclet layout schemes:
  - Layout based on funclet loading order
  - Layout based on BBT layout order
  - Layout based on pre-computed order
- One download cache for each Loader.NET component
- Multiple processes share same download cache for the same component

Into the Future

- MSIL streaming support
  - Metadata per class
  - CLR hook for class loading and release
  - Pre-JIT binary support
- Integration into platform (i.e. Fusion)
- Offline application support
  - Drizzle mode support
- Universal download cache